Target Detection Procedures and Elementary Operations for their Parallel Implementation

An Interim Technical Report for NASA Grant NAG2-1152
"Performance Characterization of Obstacle Detection Algorithms for Aircraft Navigation"
Period of the Grant: August 1, 1997 to November 30, 1998

Submitted to
NASA Ames Research Center
Technical Officer: Dr. Leland Stone
Mail Stop: 262-2
Moffett Field, California 94035

by

Rangachar Kasturi and Octavia Camps
Principal Investigators
Departments of Computer Science and Engineering and Electrical Engineering
The Pennsylvania State university
University Park, Pennsylvania 16802
Tel: (814) 863-4254 Fax: (814) 865-3176
E-Mail: {kasturi, camps}@cse.psu.edu

Graduate Students:
Tarak Gandhi
Kerry Hartman
Mau-Tsuen Yang

December 26, 1997

Target Detection Procedures, and Elementary Operations for their Parallel Implementation

Tarak Gandhi, Kerry Hartman, Mau-Tsuen Yang,
Rangachar Kasturi, and Octavia Camps

In this writeup, we have described the procedures which could be useful in target detection. We have also listed the elementary operations needed to implement these procedures. These operations could also be useful for other target detection methods. All of these operations have a high degree of parallelism, and it should be possible to implement them on a parallel architecture to enhance the speed of operation.

1 Preprocessing

This is the first step, which is used to subtract the background and remove clutter as much as possible.

1.1 Low-Stop Filtering

- Useful in removing background in case it is uniform, or slowly varying over space.
- Involves subtraction of a low pass-filtered image from the original image (or another low-pass filtered image with a smaller size, i.e. higher cut-off frequency).
- Low pass filter can be implemented by hierarchical (pyramid) method [3] as follows:
- The original image is denoted by $f(x, y)$ where (x, y) are the pixel coordinates. This forms the level 0 of the pyramid, denoted by:

$$f^{(0)} = f \tag{1}$$

The level $k+1$ is obtained from level k by low-pass filter h followed by down-sampling.

$$f^{(k+1)} = (\downarrow 2)(h * f^{(k)}) \tag{2}$$

At a certain level $k = n$ the process is stopped. The number n determines the size of the low-pass filter.

$$g^n = f^n \tag{3}$$

Reverse process is carried to obtain level $k-1$ from level k by up-sampling, and filtering by another filter h'

$$g^{(k-1)} = h' * ((\uparrow 2)g^{(k)}) \tag{4}$$

The low-pass filtered output is given by

$$g = g^{(0)} \tag{5}$$

- Spatial Convolution with 2-D filter is given by:

$$(f * h)(x, y) = \sum_{x',y'} [f(x - x', y - y') + h(x, y)] \tag{6}$$

The convolution is usually separable into two 1-D convolutions h_x and h_y where $h = h_x * h_y$.

$$(f * h_x * h_y)(x, y) = \sum_{y'} \left[\sum_{x'} [f(x - x', y - y') + h_x(x)] + h_y(y') \right] \tag{7}$$

The filter h used for low-pass filtering is given by:

$$h_x(x) = [h_x(-1)\ h_x(0)\ h_x(1)\ h_x(2)] = [1\ 3\ 3\ 1]/8 \tag{8}$$

The same filter is used for $h_y(y)$. h' is given by reflection of h; i.e. $h(x, y) = h(-x, -y)$ and is also separable.

- Down-sampling and up-sampling are given by:

$$((\downarrow 2)f)(x, y) = f(2x, 2y) \tag{9}$$

$$((\uparrow 2)f)(x, y) = f(x/2, y/2) \text{ for even } x, y \text{ ; } 0 \text{ otherwise} \tag{10}$$

- More efficient implementations are possible by doing down-sampling before low-pass, and up-sampling after high-pass. These are known as polyphase implementations [5].

1.2 Morphological Filtering

- Useful in removing large-sized clutter from small-sized objects.
- Involves subtraction between an image and its morphological opening or closing.

$$\begin{aligned} f_o &= f - (f \circ m) \\ f_c &= (f \bullet m) - f \end{aligned} \tag{11}$$

- Opening is an erosion followed by dilation, and closing is a dilation followed by erosion.

$$\begin{aligned} (f \circ m) &= (f \ominus m) \oplus m \\ (f \bullet m) &= (f \oplus m) \ominus m \end{aligned} \tag{12}$$

- Dilation and erosion are defined by [4]:

$$(f \oplus m)(x,y) = \max_{x',y'}\{f(x-x', y-y') + m(x',y')\}$$
$$(f \ominus m)(x,y) = \min_{x',y'}\{f(x+x', y+y') - m(x',y')\} \quad (13)$$

  where the values of f and m are assumed to be $-\infty$ outside the region of interest in order to make these terms redundant in max or min operations.

- The mask size assumed at present is 5×5 but slightly bigger size may be needed.

- If the mask m is separable in dimension, i.e. $m = m_x \oplus m_y$ the 2-D mask would be replaced by two 1-D masks, applied one after another.

$$(f \oplus m)(x,y) = (f \oplus m_x) \oplus m_y = \max_{y'}\{\max_{x'}\{f(x-x', y-y') + m_x(x')\} + m_y(y')\} \quad (14)$$

$$(f \ominus m)(x,y) = (f \ominus m_x) \ominus m_y = \min_{y'}\{\min_{x'}\{f(x+x', y+y') - m_x(x')\} - m_y(y')\} \quad (15)$$

- If $m(x',y') = 0$ for the whole mask i.e. region of interest, the operations of erosion and dilation are reduced to max and min operations over the mask.

2 Temporal Integration

This is useful to enhance targets with low signal to noise ratio. We have studied three approaches for performing temporal integration. These can be applied one after another in the given order.

2.1 Temporal Averaging

- Initially, a number of frames would be averaged (or summed) to bring the maximum target image velocity to one pixel per number of summed frames.

- A forgetting factor may be used to give larger weightage to more recent frames.

- May be implemented in recursive or hierarchical fashion.

Recursive implementation:

$$F(x,y;0) = 0 \quad (16)$$

$$F(x,y;t) = f(x,y;t) + \alpha F(x,y;t-1) \quad (17)$$

or

$$F(x,y;t) = (1-\alpha) f(x,y;t) + \alpha F(x,y;t-1) \quad (18)$$

where

- $f(x,y;t)$ is the value of pixel (x,y) in frame t

- $F(x, y; t)$ is the output of the algorithm at frame t obtained using the output $F(x, y; t-1)$ from the previous frame.
- α is the forgetting factor between 0 (full forgetting) and 1 (no forgetting).

Hierarchical implementation:

- Level 0 represents the original image with $t_0 = t$ as the frame number:

$$f_0(x, y; t_0) = f(x, y, t) \tag{19}$$

- Level 1 is formed by summing two consecutive images. The frame rate at level 1 is reduced by half, and the frame number is denoted by $t_1 = t_0/2$ where only even t_0 is used.

$$f_1(x, y; t_1) = f_0(x, y; t_0 + 1) + \alpha f_0(x, y; t_0) \tag{20}$$

- Level k is formed using level $k-1$ as:

$$f_k(x, y; t_k) = f_{k-1}(x, y; t_{k-1} + 1) + \alpha^k f_{k-1}(x, y; t_{k-1}) \; ; \; t_k = t_{k-1}/2 \; , \; t_{k-1} \text{ even} \tag{21}$$

  This expression is equivalent to the weighted sum of 2^k image frames, from equation (17) for $t_k = 1$ and $t = 2^k$.

2.2 Temporal Shift and Add

- This is the generalization of the hierarchical procedure for temporal averaging to account for the target image velocity.
- For simplicity, we consider the target velocity to lie between 0 and 1 pixel per frame in x and y direction. Procedure for negative pixel velocity would be the mirror image of this.
- At each hierarchical level k, the velocity can be resolved into 2^k sub-intervals per dimension, each called (u_k, v_k) representing the velocity around $(u, v) = (u_k, v_k)/2^k$. Note that these sub-intervals are not mutually exclusive, but overlap each other.
- The frame rate is reduced by half at each stage, so that the frame number $t_k = t/2^k$ for t divisible by 2^k.
- Hence, we have $2^k \times 2^k$ images corresponding to velocities from 0 to 1 in each dimension. The actual number is 4 times this, to account for negative velocities in each dimension. A pixel (x, y) in image (u_k, v_k) representing velocity around $(u_k, v_k)/2^k$, at frame t_k is denoted by $f_k(x, y; u_k, v_k; t)$.
- Images at level k are formed by shifting appropriately, and adding images at level $k-1$ according to the following equation:

$$f_k(x, y; u_k, v_k; t_k) = f_{k-1}(x, y; u_{k-1}, v_{k-1}; t_{k-1}+1) + \alpha^k f_{k-1}(x - u'_{k-1}, y - v'_{k-1}; u_{k-1}, v_{k-1}; t_{k-1}) \tag{22}$$

where

$$\begin{aligned}(u_{k-1}, v_{k-1}) &= (\lfloor u_k/2 \rfloor, \lfloor v_k/2 \rfloor) \\ (u'_{k-1}, v'_{k-1}) &= (\lceil u_k/2 \rceil, \lceil v_k/2 \rceil)\end{aligned} \tag{23}$$

- At each level, the frame rate reduces by half, but the number of states increases 4 times. Hence, the memory and computational complexity increases rapidly with the number of frames integrated and the process should be stopped before there is a resource crunch.
- However, if sub-pixel velocity resolution is required, or the SNR is low, this stage could be helpful.
- Fig. 1 shows the shift and add process for one dimension. This can be generalized to two dimensions as shown in Fig. 2.

2.3 Dynamic Programming

- To replace the temporal shift and add, after complexity has increased, dynamic programming can be used.
- Dynamic programming method was used for target detection by Barniv [2] and Arnold et al. [1].
- This process does not increase the number of states or complexity any further. Instead of increasing the number of states at the higher level, a 'maximum' operation is used.
- This process can be implemented recursively as:

$$F(x, y; u_n, v_n; 0) = 0 \tag{24}$$

$$F(x, y; u_n, v_n; t_n) = f(x, y; u_n, v_n; t_n) + \alpha^n \max_{i=0}^{\pm 1} \max_{j=0}^{\pm 1} f(x - u_n - i, y - v_n - j; u_n, v_n; t_n - 1) \tag{25}$$

where $f(x, y; u_n, v_n; t_n)$ is the value of pixel (x, y) in frame t_n, $F(x, y; u_n, v_n; t_n)$ is the output of the algorithm at frame t_n obtained using the output $F(x', y'; u_n, v_n; t - 1)$ from the previous frame, and range ± 1 is applied, so that u_n and i as well as v_n and j have same sign.

3 Spatial Integration

- All the above processes can be carried out at various levels of resolution to enable efficient detection of targets of all sizes.
- The process involves convolution and down-sampling as performed in the hierarchical low-pass filter described in Sec. 1.1.

4 Elementary Operations

The following elementary operations would be needed for the implementation of the above procedures, as well as some more algorithms.

1. Spatial Convolution with 2-D or 1-D filter: This is useful for low-pass filtering and other linear filtering applications. The equations are given in Sec. 1.1

2. Morphological dilation and erosion with 2-D or 1-D masks: This is useful for the morphological filtering to remove clutter. Maximum operator in dynamic programming can also be implemented as a special case of morphological dilation. The equations are given in Sec. 1.2.

3. Spatio-Temporal Convolution: This is the convolution performed across two dimensions of space and one of time. In most cases, this would be separable into 3 parts: This can be useful for computing optical flow using spatial and temporal gradients.

$$(f * h)(x, y; t) = \sum_{x', y', t'} [f(x - x', y - y', t - t')h(x', y', t')] \tag{26}$$

   In separable case, this becomes:

$$\begin{aligned}(f * h)(x, y; t) &= (f * h_x * h_y * h_t)(x, y, t) \\ &= \sum_{t'} \left[\sum_{y'} \left[\sum_{x'} [f(x - x', y - y', t - t')h_x(x')]\, h_y(y') \right] h_t(t') \right]\end{aligned} \tag{27}$$

4. Pointwise unary operations: This is a pointwise function of a single image. No neighboring pixels are used to perform the operation. Examples of such operations include scaling, square, square-root, etc. These could be useful for many different algorithms.

$$g(x, y) = A(f(x, y)) \tag{28}$$

5. Pointwise binary operations: This is a pointwise binary function of corresponding pixels in two images. Examples of these operations include add, subtract, multiply, divide, max, min, etc.

$$g(x, y) = B(f_1(x, y), f_2(x, y)) \tag{29}$$

6. Image warping: This is used to transform an image from one coordinate system to another. It is based on the coordinate transformation:

$$(x', y') = T(x, y) \text{ or } (x, y) = T^{-1}(x', y') \tag{30}$$

   where (x, y) and (x', y') denote the coordinates in the original and the transformed image. If T^{-1} is an integer mapping, we have:

$$g(x', y') = f(T^{-1}(x', y')) \tag{31}$$

   If T^{-1} is a real mapping, we can use bilinear transformation to interpolate the value of $g(x', y')$.

This list of operations is not exhaustive; we may come across more operations as we explore methods for target detection.

References

[1] J. Arnold, S. Shaw, and H. Pasternack. Efficient target tracking using dynamic programming. *IEEE Trans. on Aerospace and Electronic Systems*, 29(1):44–56, January 1993.

[2] Y. Barniv. Dynamic programming solution for detecting dim moving targets. *IEEE Trans. on Aerospace and Electronic Systems*, 21(1):44–56, January 1985.

[3] P. J. Burt. Fast filter transforms for image processing. *Computer Vision, Graphics and Image Processing*, 16:20–51, 1981.

[4] R. C. Gonzalez and R. C. Woods. *Digital Image Processing.* Addison-Wesley, Reading, MA, 1992.

[5] G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Wellesley, MA, 1996.

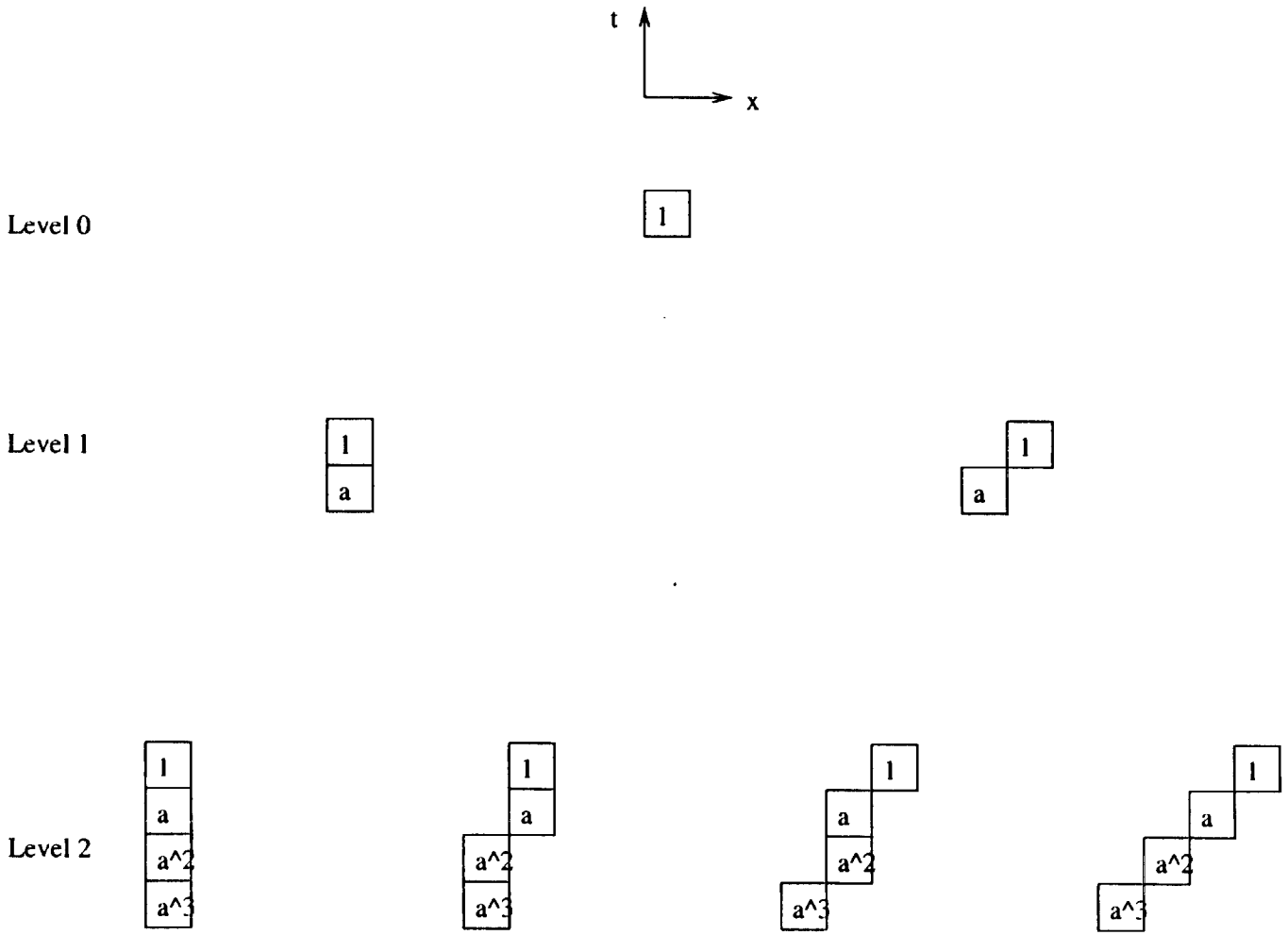

Figure 1: Shows the process of hierarchical shifting and adding of image frames in one dimension. Level 0 includes single frames. At level 1, two frames are shifted appropriately, and added with a forgetting factor a. Two velocity states are produced. At level 2, the states from level 1 are added after shifting, with a forgetting factor of a^2. This gives a forgetting factor of a if we consider the individual frames. Four velocity states are obtained.

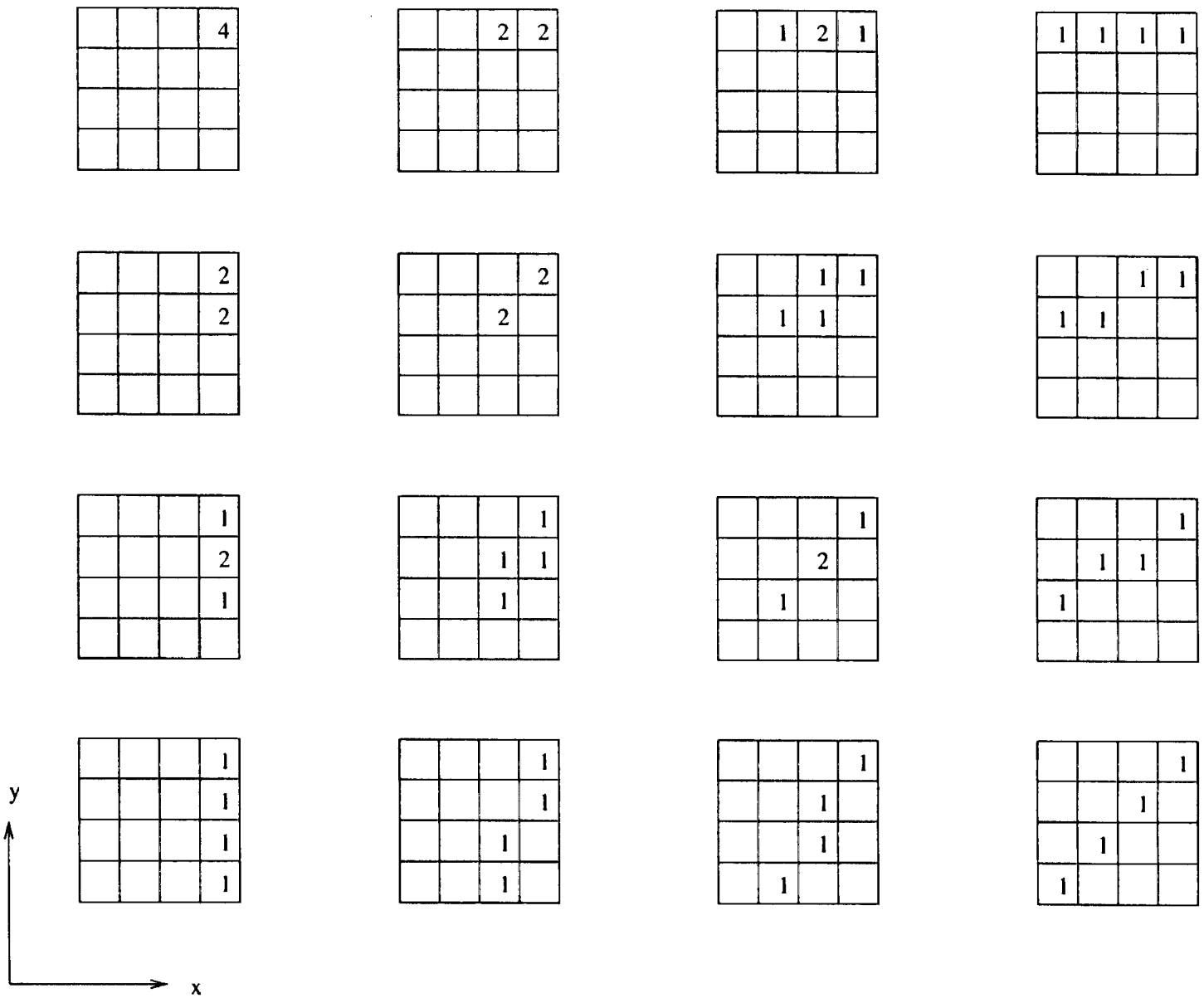

Figure 2: Shows the states obtained in case of 2-D shift and add. The pixel (x,y) is the top-right pixel, and other pixels are displaced with respect to it. In each pixel, the number denotes the number of frames added. The last frame is in pixel (x,y) and other frames as one goes backwards are along the line, corresponding to the trajectory of the target.